From Manga to Millions: What Makes Anime Films Break the Box Office

Yingshi Huang

School of Education & Information Studies
University of California, Los Angeles

1 Introduction

In September, *Demon Slayer: Kimetsu no Yaiba – The Movie: Infinity Castle* was released in North America and immediately broke box office records, generating more than $70 million in its opening weekend. It became the highest-opening anime title in North American box office history (Mcclintock, 2025). This milestone highlights not only the rising commercial power of anime films in global markets, but also the increasing cultural relevance of Japanese animation outside its country of origin. Anime films represent a significant segment of the theatrical market. According to the Motion Picture Producers Association of Japan, three of the top five films exceeding 10 billion yen in box office revenue in 2024 were anime titles: *Detective Conan: The Million-Dollar Pentagram*, *Haikyu!! The Dumpster Battle*, and *SPY × FAMILY CODE: White* (MPAL, 2024). For the North American market, industry analyses further document the rapid expansion of the anime market. The United States Anime Market: Industry Trends and Forecast 2025-2033 report notes the United States Anime Market is anticipated to grow at a compound annual growth rate of 11.49% from 2025 to 2033 to reach US $6.94 trillion by 2033 from US $2.61 billion in 2024 (Renub Research, 2025).

Motivating by the huge success of anime films, in this project, I construct a combined dataset by scraping information from MyAnimeList (MAL) and merging it with box office data from Box Office Mojo. Using these data, I focus on addressing two primary research questions: (1) How do anime films perform in the Japanese and North American theatrical markets? and (2) What factors are associated with opening-week box office performance?

2 Methodology

2.1 Data Resources

This study draws on two primary data sources: MyAnimeList (MAL) and Box Office Mojo.

MyAnimeList is a large English-language anime and manga cataloging and social networking platform with a predominantly international user base rather than one centered in Japan. Traffic analytics indicate that approximately 32.7% of visitors originate from North America, with the United States alone contributing about 29% of total website traffic (SimilarWeb, 2025). Other major user locations include Indonesia (5%), the United Kingdom (4%), Germany (3.8%), and Canada (3.7%), while more than half of the total traffic is distributed across a wide range of countries. The platform's demographic profile is largely composed of users aged 18–24, with an estimated gender distribution of roughly 70% male and 30% female. Because MAL provides structured and user-generated information on a large number of anime titles, it serves as the foundation for defining the universe of films included in this analysis. Using the Jikan API, I retrieved the top 1,000 ranked anime movies listed on MAL. For each film, metadata were collected, including the Japanese and English titles, genre classifications assigned by MAL (such as Action, Comedy, or Drama), and the MAL user score, which is computed as a weighted average designed to reduce upward bias for titles with small numbers of ratings. Additional variables include the number of users who added the film to their MAL list (a measure of audience engagement), the film's source material (e.g., original for the film, or adapted from manga, novel, etc), the age rating listed on MAL, the production studio, and the film's running time. These variables provide a rich description of both production characteristics and audience interactions.

Box Office Mojo, operated by IMDb, is a widely used industry database that systematically reports box office statistics across global and national markets (IMDb, 2025). It contains detailed financial performance data, making it well suited for examining theatrical market outcomes. For each anime title obtained from MAL, I queried the Box Office Mojo search interface and used Python's `difflib.SequenceMatcher` to identify the closest title match when discrepancies occurred between Japanese, English, or localized naming conventions. After selecting the best-matching entry, I scraped the "Performance" tables using `pandas.read_html` to extract information on the Japanese and North American (i.e., Domestic) markets, including each title's release date and opening-week box office revenue. This study focuses on opening-week rather than total gross revenue because anime films often follow heterogeneous release schedules across regions and seasons. Total revenue can be heavily affected by differences in distribution length and marketing exposure, whereas opening-week performance offers a more comparable indicator of initial audience demand and reduces bias arising from unequal theatrical run durations. All data scrapping codes were written in Python and can be found in the Appendix.

After matching opening-week box office records from Box Office Mojo to the corresponding anime titles retrieved from MAL, a total of 553 films were initially identified. To ensure the accuracy of the matched entries based on searching interface, I manually reviewed all search results and removed 22 cases in which the Box Office Mojo records did not correspond to the correct films. I then excluded titles lacking opening-week revenue data. The final analytic dataset consists of 434 films, including 328 released in the Japanese market and 106 released in the North American market. It is important to note that the Japanese and North American markets are not mutually exclusive: many anime films are released in both regions.

2.2 Statistical Analysis Procedure

The analytical strategy consists of three components: an exploratory data analysis (EDA), Nadaraya–Watson kernel regression, and a generalized additive model (GAM). The exploratory analysis provides an overview of the key variables, with particular attention to the distributional characteristics of opening-week revenue, user scores, and the number of MyAnimeList members associated with each film. These descriptive patterns establish the empirical context for subsequent modeling.

To examine the bivariate relationships between opening-week revenue and measures of audience engagement, I employ the Nadaraya–Watson kernel regression estimator. This nonparametric approach allows the data to reveal potential linear or nonlinear patterns without imposing a parametric functional form, making it well suited for detecting curvature or threshold effects in the associations between opening revenue, user score, and member count.

Building on these insights, the generalized additive model is used to jointly estimate the effects of multiple predictors while allowing for a flexible combination of linear and nonlinear terms. The GAM framework enables the inclusion of smooth functions for variables such as user score and member count, while simultaneously controlling for film-level covariates including rating category, source type, genre indicators, and release period. This approach provides a comprehensive assessment of the extent to which each factor is associated with opening-week box office performance after adjusting for other relevant characteristics.

The data analysis codes were written in both R and C, and are provided in the Appendix.

3 Results

3.1 Exploratory Data Analysis Results

As shown in Tables 1 and 2, the distribution of release years differs across markets. In the Japanese market, the largest concentration of anime film releases occurred between 2015 and 2020, whereas in the North American market the majority of releases fell between 2020 and 2025. In both regions, films adapted from manga constitute the largest category of source material, followed by original works. The most common genres are also similar across markets: Action films are the most frequent, with Adventure titles appearing second most often. Additionally, 20 films in the Japanese sample and 17 films in the North American sample received at least one award. The distribution of rating classifications is comparable across markets, with the majority of films falling under the PG-13 ("Teens 13 or older") category. For films released in Japan, the mean opening-week box office revenue is approximately $2.74 million, with an average user score of 7.56

and an average of about 99,144 members who added the film to their MyAnimeList profile. For the North American market, the corresponding averages are approximately \$2.64 million for opening-week revenue, a mean user score of 7.67, and an average of about 174,859 members.

Figure 1 presents the top and bottom three films in terms of opening-week box office performance for both the Japanese and North American markets. Across both regions, *Demon Slayer* stands out as a clear outlier, exhibiting exceptionally strong performance and far exceeding the revenues of other titles. This pattern highlights the franchise's status as a contemporary phenomenon, with unprecedented commercial reach in multiple markets. The second-highest performers include films from long-running and well-established franchises such as the *Detective Conan* series and *Dragon Ball Super*. This is notable given the age of their source materials: *Detective Conan* originated as a manga serialized in *Weekly Shōnen Sunday* beginning in 1994, and *Dragon Ball* was serialized in *Weekly Shōnen Jump* between 1984 and 1995. The sustained box office success of these adaptations suggests that classic manga properties continue to retain substantial cultural relevance and commercial viability, even decades after their initial publication. The lower-end performers also reveal a meaningful pattern. In the North American market, two of the three lowest-opening titles were released during the COVID-19 pandemic period, indicating a likely disruption in audience attendance and theatrical distribution. As noted in the descriptive summaries, the box office performance of anime films varies substantially across titles, with several extreme high-opening outliers. A similar pattern is observed for the *Members* variable, which reflects the number of MyAnimeList users who added a film to their personal list. To reduce skewness and facilitate meaningful comparisons, both variables were log-transformed prior to analysis.

The kernel density estimates of log-transformed opening-week revenue, user scores, and log-transformed member counts are displayed in Figure 2. It can be seen that the log transformation produces a more symmetric distribution of opening-week revenue in both the Japanese and North American markets. User scores, by contrast, exhibit a compressed range, with most films receiving ratings around 7.5 and only a small number exceeding a score of 9. The distributions of member counts differ more noticeably across markets. In the Japanese market, the log-members distribution appears bimodal, indicating the presence of two distinct clusters of audience engagement. In the North American market, however, most films exhibit relatively large member counts. This pattern likely reflects a selection mechanism: titles that secure theatrical releases in North America tend to be those with substantial pre-existing international fanbases, which may introduce a form of selection bias when comparing engagement metrics across regions.

The box plots of opening-week box office revenue by genre are presented in Figure 3. Across both the Japanese and North American markets, films categorized as Action, Adventure, Supernatural, Fantasy, Sports, and Comedy tend to exhibit comparatively high opening-week performance. The concentration of higher medians and wider upper ranges within these genres suggests that they consistently attract larger initial audiences, reflecting their enduring popularity and strong commercial viability in theatrical releases.

3.2 Kernel Regression Results

The user scores and member counts from MAL can be viewed as proxies for the preferences and engagement levels of the predominantly North American-based online anime fan community. Given this, it is natural to examine whether these audience indicators are associated with opening-week box office performance in the North American market, and whether they offer any predictive value for the Japanese market as well.

Nadaraya-Watson kernel regressions were estimated using a bandwidth selected via Silverman's rule of thumb. The analysis focuses on two key questions: (1) To what extent do MAL user ratings relate to opening-week box office performance in the Japanese and North American markets? and (2) To what extent do MAL member counts relate to opening-week box office performance in the two markets? The estimated regression curves, along with 95% bootstrap confidence intervals, are displayed in Figures 4 and 5.

The relationship between MAL user scores and opening-week box office differs across markets. In the North American market, the regression curve suggests an approximately linear and positive association, indicating that films with higher user ratings tend to achieve stronger openings. In contrast, for the Japanese market, MAL user scores show no clear systematic relationship with opening performance, consistent with the fact that MAL's user base largely reflects international, rather than domestic Japanese, preferences.

The relationship between member counts and opening-week revenue appears weak in both markets. The kernel regression curves show no strong or consistent association between the number of MAL users who added a film to their list and its eventual box office performance. This suggests that, even when a film has accumulated a large online following, this digital engagement does not necessarily translate into theatrical attendance.

3.3 Generalized Additive Model Results

Generalized additive models (GAMs) were fitted separately for the Japanese and North American markets to jointly assess the effects of audience indicators, film characteristics, and release conditions on opening-week box office performance. Because the exploratory analyses suggested a strong COVID-related decline in the North American market, a binary COVID indicator was included, coded as 1 for films released during 2019–2020. To avoid unstable estimates due to sparse observations, only the major genre categories, Action, Adventure, Supernatural, Fantasy, Sports, Comedy, and Award-winning, were retained, with each coded as 1 if the film belonged to that category. Source type and rating classification were also included, using *Original* and *All Ages* as the respective reference groups. Informed by the kernel regression results, the GAM specification for the North American market included a linear effect for *Score* and a nonlinear smooth term for *Members*. For the Japanese market, both *Score* and *Members* were modeled using smooth functions to capture potential nonlinearities.

The estimated coefficients and smooth terms are reported in Tables 3 and 4. For the North American market, the GAM results show that user score has a statistically significant positive association with opening-week revenue, consistent with the pattern observed in the kernel regression. The COVID indicator exhibits a strong and significant negative effect: films released during the COVID period had opening revenues approximately 79% lower than those released outside the period, calculated as $(e^{-1.55} - 1) \times 100$. Manga adaptations also performed substantially better than original works, with an estimated opening that is 232% higher. Among the genre categories, Action films display the largest effect, with opening-week revenue approximately 304% higher than that of non-Action films, highlighting the commercial dominance of this genre in the North American theatrical market.

For the Japanese market, the nonlinear smooth term for Score is significant and positive, indicating that even though MAL users are predominantly international, higher global ratings are still associated with stronger domestic openings. Manga adaptations again show significantly higher openings about 73% above those of original films. Unlike in North America, the COVID indicator does not exhibit a significant negative effect, suggesting that the Japanese theatrical market was relatively less disrupted in terms of anime film openings during the pandemic period. But the rating classifications display a significant pattern: compared with All-Ages films, PG–Children films have opening-week revenues approximately 201% higher, whereas several other rating categories show significantly lower openings. In addition, Action and Adventure films exhibit significantly stronger performance than films outside these genres, underscoring the importance of high-intensity, youth-oriented content in the Japanese market.

4 Discussion

Using data from MyAnimeList and Box Office Mojo, this study examined the performance of anime films in the Japanese and North American theatrical markets and investigated the factors associated with opening-week box office revenues. The results reveal several consistent patterns across the two markets. Notably, long-standing and classic franchises such as *Dragon Ball*, *Detective Conan*, and *Pokémon* continue to command substantial market shares, demonstrating the enduring commercial strength of legacy intellectual properties. International user ratings from MAL show meaningful associations with opening performance, particularly in the North American market, and to a lesser extent in Japan. This suggests that global fan evaluation may serve as a partial indicator of theatrical demand. Furthermore, films adapted from manga tend to perform significantly better than original works, likely reflecting the benefits of an established fanbase and pre-existing narrative recognition. Action-oriented films also exhibit consistently stronger opening revenues in both regions, underscoring the genre's broad appeal and its central role in the anime film industry.

Taken together, these findings provide useful evidence for understanding the drivers of anime box office performance and may offer practical insights for distributors and production committees in planning release strategies and marketing campaigns. For instance, incorporating international audience engagement metrics into pre-release evaluation could improve market forecasting, and prioritizing theatrical releases for franchises with strong online fanbases may yield more efficient allocation of distribution resources.

Despite these contributions, several limitations warrant consideration. Although the dataset includes information on production studios, the present analysis did not explicitly model studio-level effects. This may be a nontrivial omission: major studios such as Toei Animation appear repeatedly in both markets, contributing to 15 films in North American market and 43 films in Japanese market. Such concentration could introduce unobserved dependencies, as films produced by the same studio may share stylistic features, marketing capacity, or production quality that influence box office outcomes. In addition, the analyses conducted here are correlational in nature. While regression-based approaches can identify patterns of association, they do not establish causal relationships. Future research could incorporate more rigorous causal inference methods, such as instrumental variables, to better isolate the mechanisms through which audience engagement, source material, and genre characteristics influence box office performance.

References

IMDb. (2025). Box office mojo by imdbpro. https://www.boxofficemojo.com/

Mcclintock, P. (2025). 'demon slayer: Infinity castle' breaks anime box office opening record in north america. https://www.hollywoodreporter.com/movies/movie-news/demon-slayer-infinity-castle-record-box-office-opening-1236370035/

MPAL. (2024). *Box office statistics: 2024 annual ranking*. Motion Picture Producers Association of Japan. https://www.eiren.org/boxoffice_e/index.html

Renub Research. (2025). *United states anime market: Industry trends & forecast 2025–2033* (Report ID: 6116407). ResearchAndMarkets. https://www.researchandmarkets.com/reports/6116407

SimilarWeb. (2025). Myanimelist traffic and demographics [Accessed: 2025-01-10]. https://www.similarweb.com/website/myanimelist.net/

5 Tables

Table 1: Frequency of Year Ranges, Source Categories, Genres, and Ratings (Japanese market).

Year Range	Count	Source	Count	Genre	Count	Rating	Count
2000–2005	16	Game	28	Action	128	G – All Ages	43
2005–2010	61	Manga	147	Fantasy	90	PG-13 – Teens 13 or older	169
2010–2015	89	Novel	45	Comedy	91	PG – Children	51
2015–2020	114	Original	89	Drama	77	R – 17+ (violence & profanity)	50
2020–2025	95	Other	2	Sci-Fi	79	R+ – Mild Nudity	15
		Unknown	17	Suspense	19		
				Mystery	54		
				Supernatural	31		
				Sports	18		
				Adventure	104		
				Romance	35		
				Boys Love	2		
				Slice of Life	10		
				Horror	11		
				Avant Garde	0		
				Ecchi	4		
				Gourmet	1		
				Award Winning	20		

Table 2: Frequency of Year Ranges, Source Categories, Genres, and Ratings (North American market).

Year Range	Count	Source	Count	Genre	Count	Rating	Count
Before 2000	2	Game	6	Action	49	G – All Ages	7
2000–2005	10	Manga	55	Fantasy	37	PG-13 – Teens 13 or older	68
2005–2010	4	Novel	14	Comedy	27	PG – Children	6
2010–2015	14	Original	29	Drama	34	R – 17+ (violence & profanity)	17
2015–2020	31	Other	2	Sci-Fi	21	R+ – Mild Nudity	8
2020–2025	45			Suspense	7		
				Mystery	7		
				Supernatural	13		
				Sports	4		
				Adventure	39		
				Romance	16		
				Boys Love	1		
				Slice of Life	2		
				Horror	1		
				Avant Garde	1		
				Ecchi	1		
				Gourmet	0		
				Award Winning	17		

Table 3: General additive model estimates of the North American market.

Variables	EDF/Estimates	*p*-value
Score	**1.829**	**0.001**
s(member_log)	2.737	0.063
COVID	**-1.550**	**0.033**
Source_Manga	**1.198**	**0.025**
Source_Novel	0.010	0.989
Source_Game	1.781	0.133
Source_Other	-2.214	0.170
Rating_PG - Children	2.596	0.093
Rating_PG-13 - Teens 13 or older	0.128	0.898
Rating_R - 17+ (violence & profanity)	-0.863	0.427
Rating_R+ - Mild Nudity	0.786	0.516
Duration_min	0.000	0.995
Action	**1.401**	**0.009**
Adventure	0.144	0.770
Supernatural	0.773	0.303
Fantasy	-0.035	0.948
Sports	1.757	0.126
Comedy	-0.246	0.649
Award_win	-0.881	0.162

Table 4: General additive model estimates of the Japanese market.

Variables	EDF/Estimates	*p*-value
s(Score)	**1.789**	**0.002**
s(member_log)	2.350	0.159
COVID	-0.228	0.162
Source_Manga	**0.551**	**0.000**
Source_Novel	-0.031	0.863
Source_Game	0.112	0.615
Source_Other	-0.946	0.155
Rating_PG - Children	**1.103**	**0.000**
Rating_PG-13 – Teens 13 or older	0.026	0.911
Rating_R - 17+ (violence & profanity)	**-0.773**	**0.000**
Rating_R+ - Mild Nudity	**-1.277**	**0.000**
Duration_min	**-1.470**	**0.000**
Action	**0.008**	**0.003**
Adventure	**0.410**	**0.001**
Supernatural	0.212	0.150
Fantasy	0.105	0.581
Sports	0.028	0.856
Comedy	0.378	0.116
Award_win	0.124	0.345

6 Figures

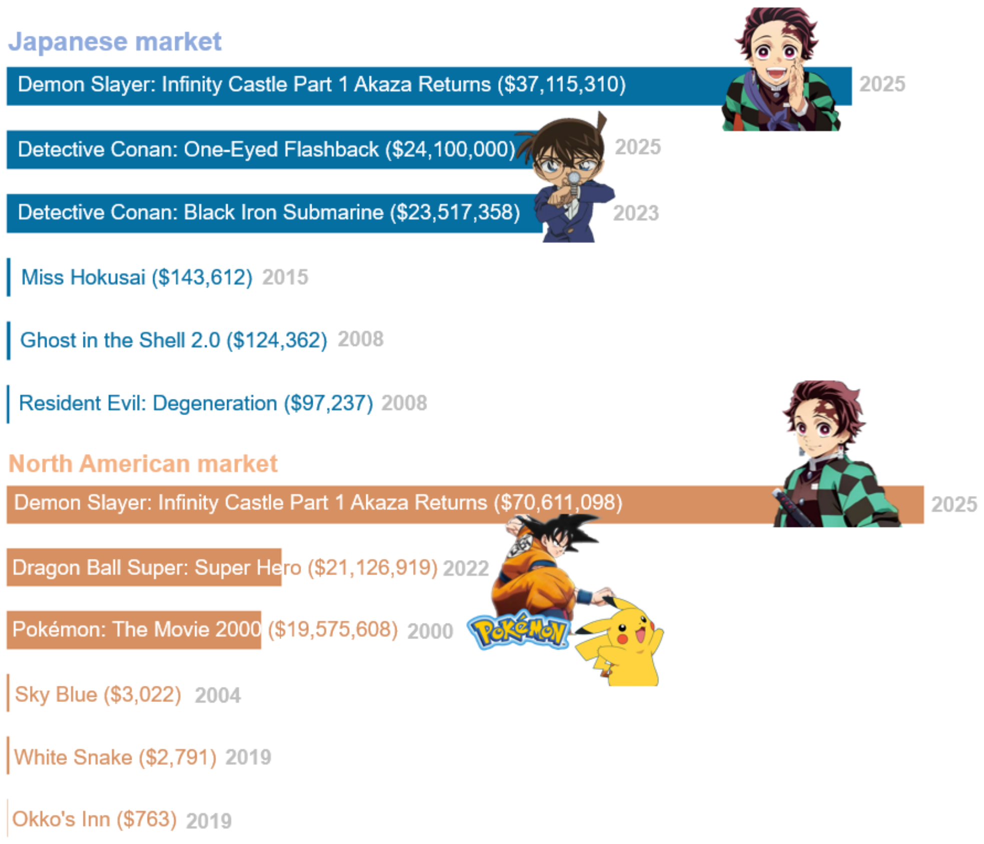
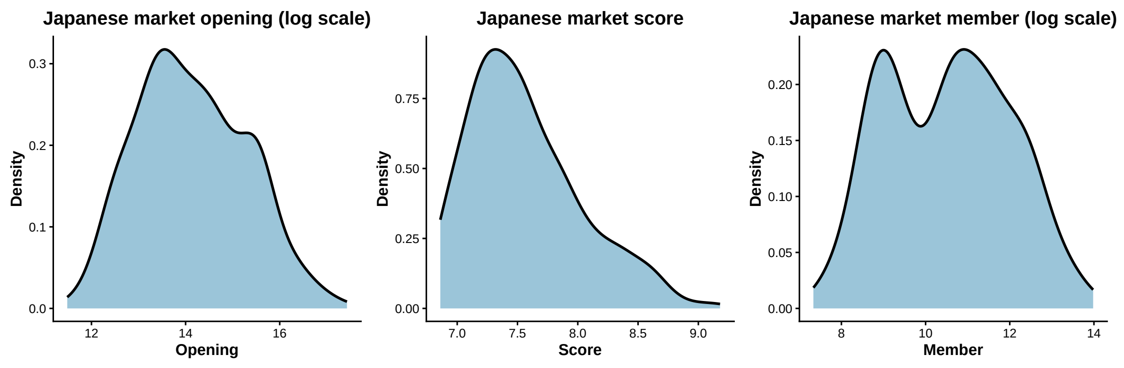
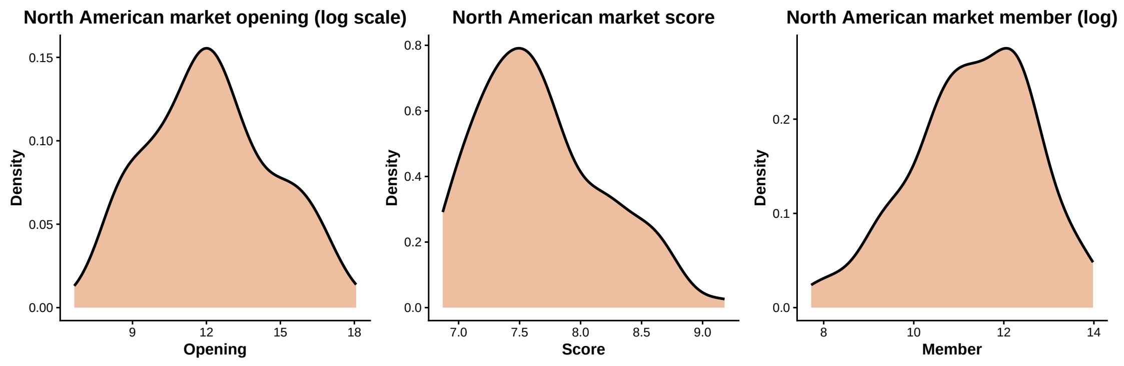


Figure 1: The top 3 and bottom 3 anime films in the Japanese market and the North American market.



(a) Japanese market



(b) North American market

Figure 2: Kernel density for opening-week box office in log scale, user scores, and the number of members in log scale for the Japanese market (a), and the North American market (b).

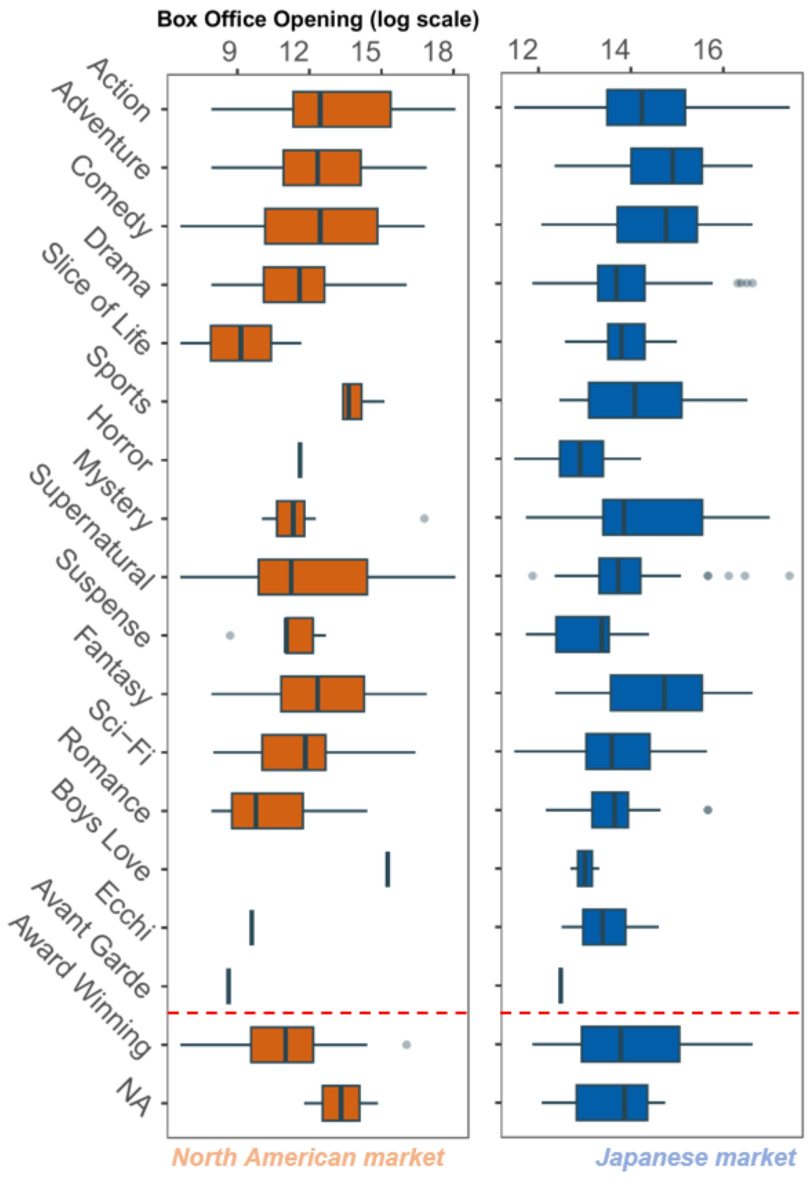
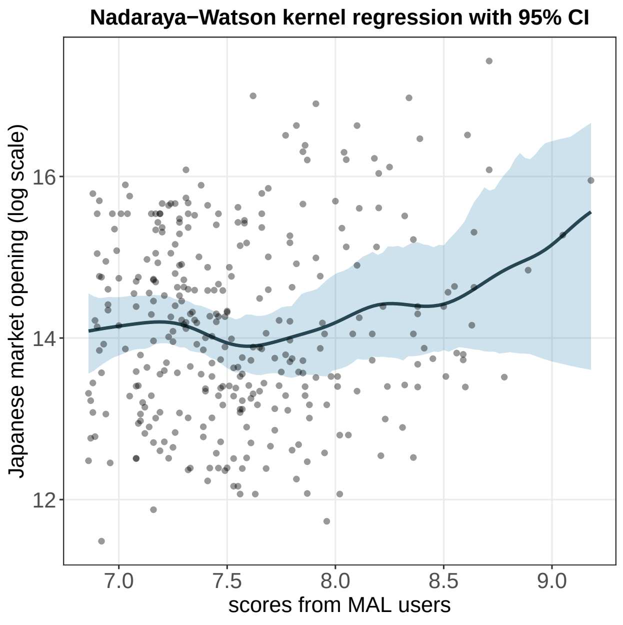
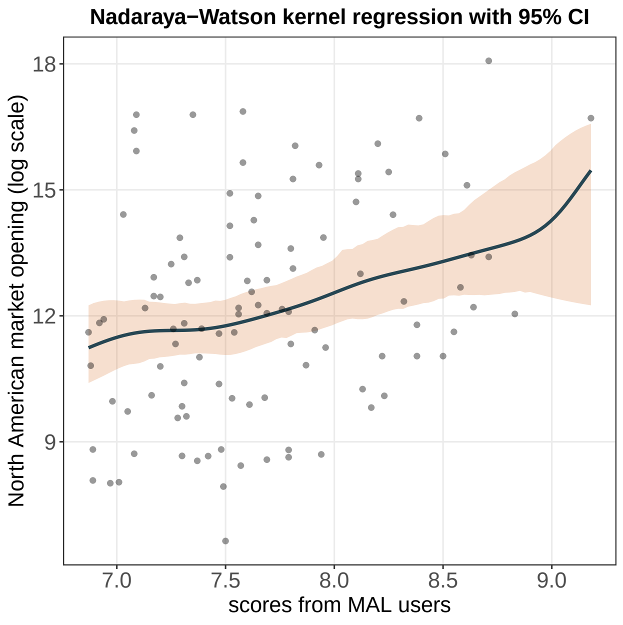


Figure 3: Box plot of log-transformed opening-week box office performance by genre for the Japanese and North American markets.

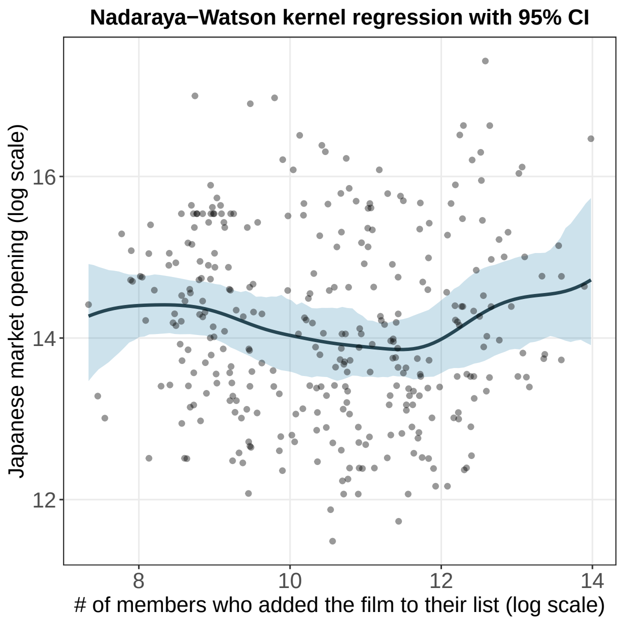


(a) Japanese market

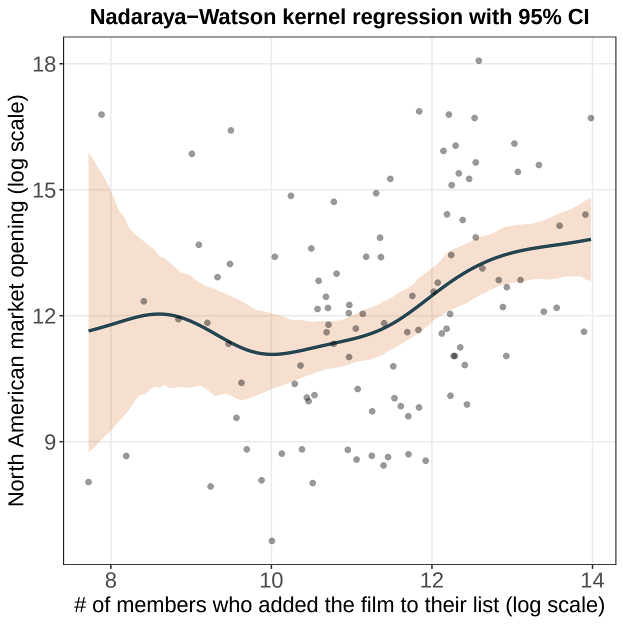


(b) North American market

Figure 4: Nadaraya–Watson kernel regression between scores from MAL users and opening in log scale.



(a) Japanese market



(b) North American market

Figure 5: Nadaraya–Watson kernel regression between members and opening in log scale.

7 Appendix

7.1 Python Codes for Data Scraping

Listing 1: Python codes used for scraping data from MyAnimeList.

```python
import requests
import time
import csv

BASE_URL = "https://api.jikan.moe/v4/top/anime"

HEADERS = {
    "User-Agent": "MyAnimeResearchBot/1.0"
}

def fetch_top_movies_page(page):
    """
    Jikan: Top Anime
    """
    params = {
        "type": "movie",
        "page": page
    }
    print(f"Fetching page {page} ...")
    resp = requests.get(BASE_URL, params=params, headers=HEADERS)
    resp.raise_for_status()
    data = resp.json()
    return data


def main():
    results = []
    for page in range(1, 41):
        data = fetch_top_movies_page(page)
        anime_list = data.get("data", [])

        for item in anime_list:
            rank = item.get("rank")
            title = item.get("title")
            type_ = item.get("type")
            episodes = item.get("episodes")
            score = item.get("score")
            scored_by = item.get("scored_by")
            popularity = item.get("popularity")
            members = item.get("members")
            status = item.get("status")
            url = item.get("url")

            source_ = item.get("source")
            duration = item.get("duration")
            rating = item.get("rating")

            year = item.get("year")

            aired = item.get("aired") or {}
            aired_from = aired.get("from")  # '2020-10-16T00:00:00+00:00'

            def join_names(key):
                arr = item.get(key) or []
                names = [x.get("name") for x in arr if x.get("name")]
                return ", ".join(names) if names else None

            genres = join_names("genres")
            themes = join_names("themes")
            demographic = join_names("demographics")
            studios = join_names("studios")

            results.append({
                "Rank": rank,
                "Title": title,
                "Genres": genres,
                "Themes": themes,
                "Demographic": demographic,
                "Status": status,
                "Type": type_,
                "Episodes": episodes,
                "Score": score,
                "Scored_By": scored_by,
                "Popularity": popularity,
                "Members": members,
                "Year": year,
                "Aired_From": aired_from,
                "Source": source_,
                "Duration": duration,
                "Rating": rating,
                "Studios": studios,
                "URL": url,
            })

        time.sleep(1)

    fieldnames = [
        "Rank",
        "Title",
        "Genres",
        "Themes",
        "Demographic",
        "Status",
        "Type",
        "Episodes",
        "Score",
        "Scored_By",
        "Popularity",
        "Members",
        "Year",
        "Aired_From",
        "Source",
        "Duration",
        "Rating",
        "Studios",
        "URL",
    ]

    with open("top_1000_anime_movies_jikan.csv", "w", newline="", encoding="utf-8-sig") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        for row in results:
            writer.writerow(row)

    print("Done! Saved to top_1000_anime_movies_jikan.csv")


if __name__ == "__main__":
    main()
```

Listing 2: Python codes used for scraping data from Box Office Mojo.

```
import requests
import time
import re
import difflib
import pandas as pd
from bs4 import BeautifulSoup
from urllib.parse import quote

# ================== setting ==================

N_MOVIES = 1000

# Jikan Top Anime
JIKAN_TOP_URL = "https://api.jikan.moe/v4/top/anime"
JIKAN_ANIME_FULL_URL = "https://api.jikan.moe/v4/anime/{mal_id}/full"

# Box Office Mojo
BOM_SEARCH_BASE = "https://www.boxofficemojo.com/search/?q="
BOM_TITLE_BASE = "https://www.boxofficemojo.com"

HEADERS = {
    "User-Agent": (
        "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
        "AppleWebKit/537.36 (KHTML, like Gecko) "
        "Chrome/120.0.0.0 Safari/537.36"
    )
}

SLEEP_JIKAN_TOP = 1.0
SLEEP_JIKAN_FULL = 1.5
SLEEP_BOM_SEARCH = 2.0
SLEEP_BOM_TITLE = 3.0

OUT_MOVIES_CSV = "jikan_top_movies_with_alt_titles.csv"
OUT_BOM_PERF_CSV = "anime_movies_boxoffice_performance_by_alt_titles.csv"


# ================== query functions ==================

def get_json(url, params=None, sleep_sec=0, max_retries=3):
    for attempt in range(max_retries):
        try:
            resp = requests.get(url, params=params, headers=HEADERS, timeout=30)
            resp.raise_for_status()
            if sleep_sec > 0:
                time.sleep(sleep_sec)
            return resp.json()
        except requests.exceptions.HTTPError as e:
            status = getattr(e.response, "status_code", None)
            print(f"[HTTPError] {status} when GET {url} (attempt {attempt+1}/{max_
                retries})")
            if status in (429, 503) and attempt < max_retries - 1:
                time.sleep(5 * (attempt + 1))
                continue
            raise
        except requests.exceptions.RequestException as e:
            print(f"[RequestException] {e} (attempt {attempt+1}/{max_retries})")
            if attempt < max_retries - 1:
                time.sleep(5 * (attempt + 1))
                continue
            raise


def get_html(url, sleep_sec=0, max_retries=3):
    for attempt in range(max_retries):
        try:
            resp = requests.get(url, headers=HEADERS, timeout=30)
            resp.raise_for_status()
            if sleep_sec > 0:
                time.sleep(sleep_sec)
            return resp.text
        except requests.exceptions.HTTPError as e:
            status = getattr(e.response, "status_code", None)
            print(f"[HTTPError] {status} when GET {url} (attempt {attempt+1}/{max_
                retries})")
            if status in (429, 503) and attempt < max_retries - 1:
                time.sleep(5 * (attempt + 1))
                continue
            raise
        except requests.exceptions.RequestException as e:
            print(f"[RequestException] {e} (attempt {attempt+1}/{max_retries})")
            if attempt < max_retries - 1:
                time.sleep(5 * (attempt + 1))
                continue
            raise


# ================== utility functions ==================

def normalize_title(s):
    if not s:
        return ""
    s = s.lower()
    s = re.sub(r"[^a-z0-9]+", " ", s)
    s = re.sub(r"\s+", " ", s).strip()
    return s


def uniq(seq):
    seen = set()
    out = []
    for x in seq:
        if x and x not in seen:
            seen.add(x)
            out.append(x)
    return out

```

```

# ================== Jikan Top Movie basic info ==================

def fetch_top_movies(max_n=N_MOVIES):
    movies = []
    page = 1

    while len(movies) < max_n:
        params = {"type": "movie", "page": page}
        print(f"[Jikan] Fetching top movies page {page} ...")
        data = get_json(JIKAN_TOP_URL, params=params, sleep_sec=SLEEP_JIKAN_TOP)
        page_data = data.get("data", [])
        if not page_data:
            print("[Jikan] No more data from top endpoint.")
            break

        for item in page_data:
            movies.append({
                "MAL_ID": item.get("mal_id"),
                "Rank": item.get("rank"),
                "Title_Main": item.get("title"),
                "Year": item.get("year"),
                "Score": item.get("score"),
                "Popularity": item.get("popularity"),
                "Members": item.get("members"),
                "Jikan_URL": item.get("url"),
                "Anime_Type": item.get("type"),
                "Episodes": item.get("episodes"),
            })
            if len(movies) >= max_n:
                break

        if len(movies) >= max_n:
            break

        page += 1

    print(f"[Jikan] Collected {len(movies)} movies from top list.")
    return movies


# ================== get Jikan full Alternative Titles ==================

def get_anime_full(mal_id):
    url = JIKAN_ANIME_FULL_URL.format(mal_id=mal_id)
    print(f"[Jikan-full] Fetching full info for MAL id {mal_id} ...")
    data = get_json(url, sleep_sec=SLEEP_JIKAN_FULL)
    return data.get("data", {})


def build_title_candidates(anime_full, base_title=None):
    candidates = []

    if anime_full.get("title_english"):
        candidates.append(anime_full["title_english"])

    for t in anime_full.get("titles") or []:
        t_type = t.get("type")
        t_title = t.get("title")
        if t_type == "English" and t_title:
            candidates.append(t_title)

    for syn in anime_full.get("title_synonyms") or []:
        candidates.append(syn)

    if base_title:
        candidates.append(base_title)

    if anime_full.get("title"):
        candidates.append(anime_full["title"])

    return uniq(candidates)


# ================== search Box Office Mojo and get imdb_id ==================

def search_bom_by_title_single(title, year=None):
    query = quote(title)
    url = f"{BOM_SEARCH_BASE}{query}"
    print(f"[BOM-SEARCH] Searching '{title}' -> {url}")
    html = get_html(url, sleep_sec=SLEEP_BOM_SEARCH)
    soup = BeautifulSoup(html, "lxml")

    candidates = []
    for a in soup.select('a[href^="/title/tt"]'):
        href = a.get("href", "")
        m = re.search(r"/title/(tt\d+)/", href)
        if not m:
            continue
        imdb_id = m.group(1)
        cand_title = a.get_text(strip=True)

        row = a.find_parent("tr")
        if row is not None:
            row_text = " ".join(row.stripped_strings)
        else:
            row_text = cand_title

        candidates.append((imdb_id, cand_title, row_text))

    if not candidates:
        print("  [BOM-SEARCH] No candidates found for this query.")
        return None, None, 0.0

    base_norm = normalize_title(title)
    best_score = -1.0
    best = None

    for imdb_id, cand_title, row_text in candidates:
        cand_norm = normalize_title(cand_title)
        title_sim = difflib.SequenceMatcher(None, base_norm, cand_norm).ratio()

        year_bonus = 0.0
        if year:
            years_in_row = re.findall(r"\b(19\d{2}|20\d{2})\b", row_text)
            if str(year) in years_in_row:
                year_bonus = 0.1

        score = title_sim + year_bonus

        if score > best_score:
            best_score = score
            best = (imdb_id, cand_title)

    if best_score < 0.5:
        print(f"  [BOM-SEARCH] Best score {best_score:.2f} < 0.5, consider as no
            reliable match.")
        return None, None, best_score

    imdb_id, best_title = best
    print(f"  [BOM-SEARCH] Best match: {best_title} ({imdb_id}), score={best_score
        :.2f}")
    return imdb_id, best_title, best_score


def search_bom_by_titles(candidates, year=None):
    overall_best = (None, None, None, 0.0)  # imdb_id, bom_title, cand_used, score

    for cand in candidates:
        imdb_id, bom_title, score = search_bom_by_title_single(cand, year=year)
        if imdb_id is None:
            continue
        if score > overall_best[3]:
            overall_best = (imdb_id, bom_title, cand, score)

    imdb_id, bom_title, cand_used, best_score = overall_best
    if imdb_id is None:
        print("  [BOM-SEARCH] No reliable match from any candidate title.")
    else:
        print(f"  [BOM-SEARCH] FINAL match: IMDb {imdb_id}, BOM title '{bom_title
            }', "
              f"via candidate '{cand_used}', score={best_score:.2f}")
    return overall_best


# ================== based on imdb_id and get Box Office Mojo Performance table
    ==================

def get_bom_performance_table(imdb_id):
    url = f"{BOM_TITLE_BASE}/title/{imdb_id}/"
    print(f"[BOM-TITLE] Fetching performance for {imdb_id} -> {url}")
    html = get_html(url, sleep_sec=SLEEP_BOM_TITLE)

    try:
        tables = pd.read_html(html)
    except ValueError:
        print("  [BOM-TITLE] No tables found on BOM page.")
        return None

    perf_tables = []
    for df in tables:
        if {"Area", "Release Date", "Opening", "Gross"}.issubset(df.columns):
            perf_tables.append(df)

    if not perf_tables:
        print("  [BOM-TITLE] No Area/Release Date/Opening/Gross table.")
        return None
```

```python

    perf_df = pd.concat(perf_tables, ignore_index=True)
    perf_df["IMDb_ID"] = imdb_id
    perf_df["BoxOfficeMojo_URL"] = url
    return perf_df


# ================== main ==================

def main():
    # 1) Jikan basic info
    movies = fetch_top_movies(max_n=N_MOVIES)
    df_movies = pd.DataFrame(movies)
    df_movies.to_csv(OUT_MOVIES_CSV, index=False, encoding="utf-8-sig")
    print(f"\n[OUTPUT] Saved Jikan base movie info to {OUT_MOVIES_CSV}")

    all_perf_dfs = []

    for idx, movie in enumerate(movies, start=1):
        mal_id = movie["MAL_ID"]
        title_main = movie.get("Title_Main")
        year = movie.get("Year")

        print(f"\n===== [{idx}/{len(movies)}] MAL {mal_id} | {title_main} ({year})
            =====")

        # 2) Jikan full: get Alternative Titles
        try:
            anime_full = get_anime_full(mal_id)
        except Exception as e:
            print(f"  [ERROR] Jikan full error for MAL {mal_id}: {e}")
            continue

        candidates = build_title_candidates(anime_full, base_title=title_main)
        print("  [INFO] Title candidates:", " | ".join(candidates))

        movie["AltTitle_Candidates"] = " | ".join(candidates)

        # 3) search Box Office Mojo with titles to get imdb_id
        try:
            imdb_id, bom_title, cand_used, match_score = search_bom_by_titles(
                candidates, year=year)
        except Exception as e:
            print(f"  [ERROR] BOM search error for MAL {mal_id}: {e}")
            continue

        if imdb_id is None:
            print("  [SKIP] No reliable BOM match, skip this movie.")
            continue

        # 4) based on imdb_id to get Performance table
        try:
            perf_df = get_bom_performance_table(imdb_id)
        except Exception as e:
            print(f"  [ERROR] BOM performance error for '{title_main}' ({imdb_id})
                : {e}")
            continue

        if perf_df is None or perf_df.empty:
            continue

        # Jikan and matching
        perf_df["Title_Jikan_Main"] = title_main
        perf_df["Year_Jikan"] = year
        perf_df["Rank_Jikan"] = movie.get("Rank")
        perf_df["Jikan_Score"] = movie.get("Score")
        perf_df["Jikan_Popularity"] = movie.get("Popularity")
        perf_df["Jikan_Members"] = movie.get("Members")
        perf_df["MAL_ID"] = mal_id
        perf_df["Jikan_URL"] = movie.get("Jikan_URL")
        perf_df["Anime_Type"] = movie.get("Anime_Type")
        perf_df["Episodes"] = movie.get("Episodes")
        perf_df["AltTitle_Candidates"] = movie.get("AltTitle_Candidates")
        perf_df["Matched_Title_Used"] = cand_used
        perf_df["BOM_Title_Matched"] = bom_title
        perf_df["Match_Score"] = match_score

        all_perf_dfs.append(perf_df)

    if not all_perf_dfs:
        print("[MAIN] No BOM performance data collected.")
        return

    final_df = pd.concat(all_perf_dfs, ignore_index=True)
    final_df.to_csv(OUT_BOM_PERF_CSV, index=False, encoding="utf-8-sig")
    print(f"\n[OUTPUT] Saved BOM performance table to {OUT_BOM_PERF_CSV}")


if __name__ == "__main__":
    main()
```

7.2 R and C Codes for Data Analyses

Listing 3: R and C codes used for statistical analyses.

```
library(Rcpp)
## 1. Kernel regression -----
## the C function for Nadaraya-Watson kernel regression
cppFunction('
NumericVector NWkernreg(NumericVector x,       // predictor
                        NumericVector y,       // dependent variable
                        NumericVector g2,      // gridpoints
                        double b               // bandwidth
                        ){
  int n = x.size();            // # of data points
  int m = g2.size();           // # of the girdpoints
  NumericVector res2(m);       // kernel regression estimates

  for(int i = 0; i < m; i++){
    double num = 0.0;          // the numerator
    double den = 0.0;          // the denominator

    for(int j = 0; j < n; j++){
      double u = (g2[i] - x[j])/b;             // standardized distance
      double w = std::exp(-0.5*pow(u, 2));     // weight for each data point

      num += w * y[j];
      den += w;
    }

    if(den > 0.0){
      res2[i] = num / den;
    }else{
      res2[i] = 0.0;
    }
  }

  return res2;
}
')

## estimation ----
# X <- subdata$Score
X <- subdata$member_log
Y <- subdata$open_log
## bandwidth using Silverman rule-of-thumb
bw_rule <- 1.06 * sd(X) * length(X)^(-1/5)

## 100 equally spaced Petrol tax
spaced_X <- seq(min(X), max(X), length = 100)

## compute the NW kernel regression estimates
fhat <- NWkernreg(x = X, y = Y, g2 = spaced_X, b = bw_rule)

nrepeat <- 200
fhat_boot <- matrix(data = NA, nrow = nrepeat,
                    ncol = length(fhat))
for (i in 1:nrepeat){
  idx <- sample(1:length(X), size = 100, replace = TRUE)
  boot_X <- X[idx]
  boot_Y <- Y[idx]

  ## compute the NW kernel regression estimates
  fhat_boot[i, ] <- NWkernreg(x = boot_X, y = boot_Y,
                              g2 = spaced_X, b = bw_rule)
}

## compute the 2.5th and 97.5th percentiles
lower <- apply(fhat_boot, 2, quantile, probs = 0.025)
upper <- apply(fhat_boot, 2, quantile, probs = 0.975)

## plotting -----
df_plot <- data.frame(
  x     = spaced_X,
  fhat  = fhat,
  lower = lower,
  upper = upper
)
raw_dat <- data.frame(X = X, Y = Y)

if(use_area == "Japan"){
  y_label <- "Japanese market opening (log scale)"
}else{
  y_label <- "North American market opening (log scale)"
}

```

```
pdf(file = "member_open_NWker_jap.pdf",
    width = 6, height = 6)
ggplot(df_plot, aes(x = x)) +
  geom_ribbon(aes(ymin = lower, ymax = upper),
              # fill = "#D36113",
              fill = "#076fa2",
              alpha = 0.2) +
  geom_line(aes(y = fhat), color = "#264653", size = 1.1) +
  geom_point(data = raw_dat, aes(x = X, y = Y),
             alpha = 0.4, size = 1.5) +
  labs(
    title = "Nadaraya-Watson kernel regression with 95% CI",
    x = "# of members who added the film to their list (log scale)",
    y = y_label
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(size = 15, face = "bold", hjust = 0.5),
    axis.title.x = element_text(size = 15),
    axis.title.y = element_text(size = 15),

    axis.text.x = element_text(size = 15),
    axis.text.y = element_text(size = 15),

    panel.grid.minor = element_blank()
  )
dev.off()

## 2. General Additive Model ------
library(mgcv)
gam_model <- gam(
  open_log ~
    s(Score) +
    s(member_log) +
    COVID +
    Source_mer +
    Rating +
    Duration_min +
    Action +
    Adventure +
    Supernatural +
    Fantasy +
    Sports +
    Comedy +
    award,
  data = subdata,
  family = gaussian()
)
summary(gam_model)
plot(gam_model, shade = TRUE, seWithMean = TRUE)
write.csv(cbind(test$p.coeff, test$p.pv), file = "jap_gam.csv")
```